SLIC implementation in Java (ImageJ)

JB

16th June 2013

Abstract

This short report describes some detail about SLIC (Simple Linear Iterative Clustering) implementation in Java as an ImageJ plug-in. Basically we re-implemented the SLIC by the [1] and publicly available C++ implementation by authors. We compare the speed and result of both implementation. Further we disuse some possibilities of speeding up the Java implementation such as used data structures, multi-threading and advance sampling.

1 Introduction

Many image processing tasks in medical imaging from a few past years started to deal with very large images which rapidly increase the problem dimensionality and the computation time which is a reason why an reduction in the way of some superpixels would be very welcome. Recently the SLIC (Simple Linear Iterative Clustering) [1] was introduced for general images and presented as a powerful tool in medical imaging [3, 2].

The computer vision problems are solved solved in many languages and platforms. The original C++ code[1] provided by authors was later on wrapped to Python[2] and Matlab in VLFeat[3] library. For a real-time computer vision problems the SLIC was also transformed with some minor improvements to be fully proceeded on graphic cards as a gSLIC [4]. We join this approach and we create an Java implementation with plug-in interface to ImageJ[4] which suppose to be one of the widely used image processing platform in medical imaging.

2 Implementation

We implemented the SLIC method in Java as an independent java class with an plugin interface to ImageJ (Fiji[5] respectively). We followed the the original

[1] `http://ivrg.epfl.ch/research/superpixels`
[2] `https://github.com/amueller/slic-python`
[3] `http://www.vlfeat.org/`
[4] `http://rsbweb.nih.gov/ij/`
[5] `http://fiji.sc/Fiji`

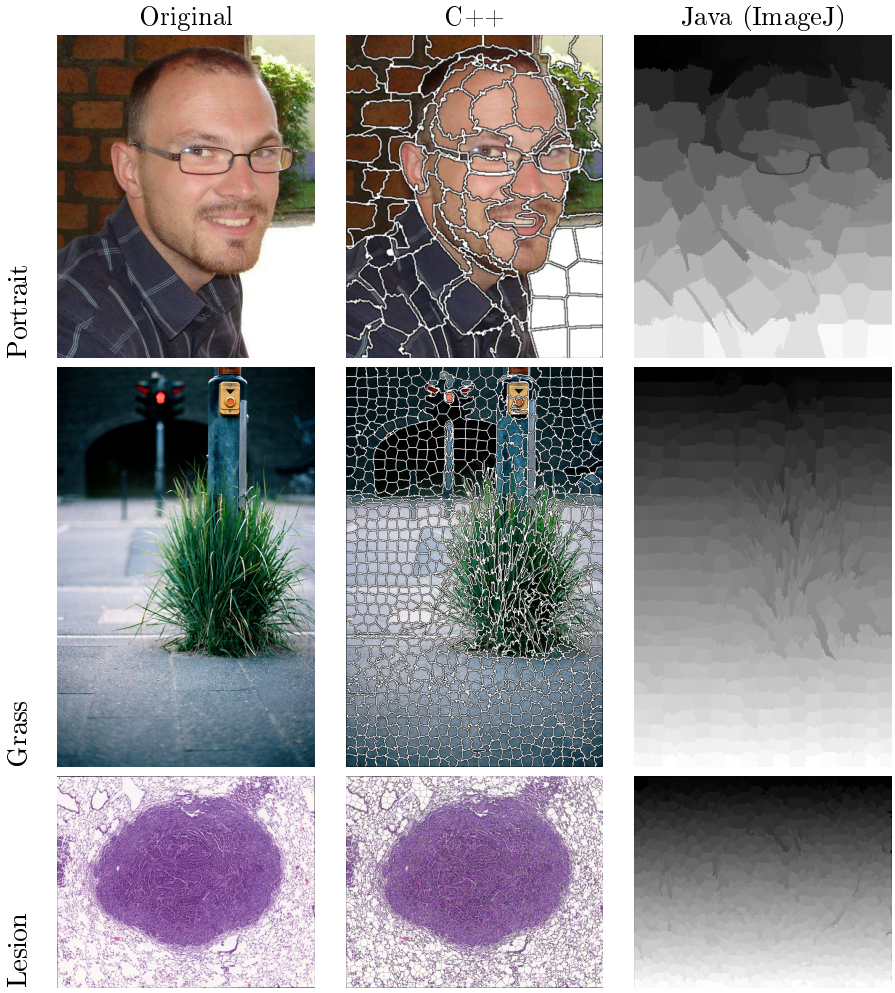

Figure 1: Sample images are presented in rows when each has different image size (Portrait has 234×445 pixels, Grass has 658×1024 pixels and Lesion has 4480×3690 pixels). The original images are shown in the first column, then the result of C++ implementation in the second column and Java (ImageJ plug-in) implementation in the last column.

Images	C++	Java (simple)
Portrait 234×445 pixels	0.55s	0.8s
Grass 658×1024 pixels	2.46s	12.4s
Lesion 4480×3690 pixels	62.6s	391s

Table 1: Comparison of segmentation time of C++ and Java implementation for three different image sizes.

method [1] and later on we took inspiration from some other implementations such as VLFeat to have regularisation in range $(0, 1)$ and independent on the image and grid size. As we speak in following sections 3 and 4 we also discuss some other improvements namely more reasonable post-processing and speeding up.

2.1 Comparison C++ vs Java

We made basic speed comparison between the C++ (we used the Python wrapper above original C++ code) and our Java implementation. We tested the superpixel segmentations with reliable parameters on a few images with different image sizes from small to large, see Fig.1.

By the comparison[6] we found that the Java is expectantly slower then the C++ implementations (difference between compiled and interpreted languages). This is the reason why we propose some speeding-ups in Sec.4.

3 Reasonable post-processing

The original implementation of SLIC contains only simple post processing step. It estimates the connected segments using region growing method depending on connectivity (the original code assumes 4-neighboring).Then if the a segment is smaller the the initial superpixel size it relabel this segment by label taken from surrounding of the first assigned pixel to this segment. By our opinion it is quite pure solution and it assumes relabelling only small segments fully surrounded by another larger segment.

We propose different approach which involves the size of this small segments (for example only segments smaller the 10% of initial superpixel size will be merged/relabels) and also similarity between neighbouring segments. This way should not be markedly larger but it should produce reasonable segmentations especially for estimating large superpixels on indented structures.

[6] Later we add the speed of the Java implementation with proposed speedups.

	Time [ms]
int[][][]	441ms
ShortProcessor	2146ms

Table 2: Comparison of reading and writing time for images of size 10.000×10.000 pixels using ImageJ ShortProcessor and simple 2D array of primitive int. The 2D array according our benchmark is about 5 times faster.

4 Other possible Speed-ups

The basis implementation of SLIC in Java could not be expected to be equally fast as the C++ implementation is because C++ is compiled to binaries and Java is still interpreted language only. For this purpose we propose a few possibilities how to speed-up the clustering process with minimal precision lose.

4.1 Native Java structures

Java contains many data structures which can be used for simpler handling complex data structures. With respect to the integration of the segmentation tool to ImageJ framework we also assumed to use the ShortProcessor[7] as the basic data structure for our segmentations. One of the main criterion is the processing speed. from a benchmark presented in tab.2 we found that using int[][][] structure is about 5 times faster then ShortProcessor.

4.2 Pre-computing colour space conversion

Because of several mathematical (multiplications and division are quite costly operation) operation per each image pixel, the colour conversion from RGB[8] to CIE-Lab[9] is coming to be quite time consuming especially for huge images (typically of size 8.000×8.000 pixels and more). If we assume that RGB image is represented by three bytes the number of possible colours is limited about $16 \cdot 10^6$ compare to $64 \cdot 10^6$ for image of size 8.000×8.000 pixels. Also we observed that in the image any real image are not all possible colours, histological images are uses less then 5% of the colour space which means less then $8 \cdot 10^5$ colours.

We suggest to compute each colour conversion only once, then store in in a temporary array and next time this colour needs to be converted we simply get it from temporary 3D array.

4.3 Multi-threading

So far the clustering is compute locally is is quite simple both phases (assignment and update) split into independent tasks and run them in threads. In

[7] `http://rsb.info.nih.gov/ij/developer/api/ij/process/ShortProcessor.html`
[8] `http://en.wikipedia.org/wiki/RGB_color_model`
[9] `http://en.wikipedia.org/wiki/Lab_color_space`

the assignment phase where the top loop is running over all clusters we process a subset of these clusters in individual threads and use shared variable for computed distances and and minimal distances to the closest cluster. The update step can be also computed per image image blocks such that each thread processes own image block.

For implementation we create only N thread defined in the ImageJ configuration.

4.4 Under-sampling for huge images

Because of the pixel distance takes quite big part of the SLIC metric (so it prefers compact segmentations) and images are defined on regular pixel grid we can proposed an under-sampling for computing the distances to given cluster and assignment the label. Another justification of this simplification is...

We propose to compute the distance and assignment for every second pixel in the image and then extend this labelling also for the neighbouring pixels according to chosen neighbouring connectivity. For images with very uniform texture and in case estimation relatively large superpixels (with respect to the image size) we could take event larger step the 2 we proposed before.

References

[1] R Achanta and A Shaji. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *Pattern Analysis and Machine Intelligence, IEEE*, 34(11):2274 – 2282, 2012.

[2] Jiri Borovec, Jan Kybic, Michal Bušta, Carlos Ortiz-de Solorzano, and Arrate Munoz-Barrutia. Registration of multiple stained histological sections. In *International Symposium on Biomedical Imaging, IEEE*, pages 1022–1025, San Francisco, 2013.

[3] A Lucchi, K Smith, and R Achanta. Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks With Learned Shape Features. *Medical Imaging, IEEE*, 31(2):474 – 486, 2012.

[4] CY Ren and Ian Reid. gSLIC: a real-time implementation of SLIC superpixel segmentation. Technical report, 2011.